17-3-2025

RETO TÉCNICO

Mercado Libre

Maximiliano Di Pauli

Contenido

Reto Técnico Night Watch Team Infraestructure 2

1. Introducción 2
2. Infraestructura en AWS 2
3. Infraestructura en Kubernetes 4
4. Seguridad y Escalabilidad 6
5. Monitoreo y Alertas 7
6. Conclusión 7
7. Documentación Utilizada 7

Reto Técnico Night Watch Team Infraestructure

1. Introducción

Este documento describe la implementación de una infraestructura escalable y segura utilizando AWS y Kubernetes como parte del Reto Tecnico Night Watch Team Infraestructura.

Para la gestión y despliegue de la infraestructura, se han utilizado herramientas de Infraestructura como Código (IaC), específicamente Terraform para la provisión de recursos en AWS y Kubernetes para la orquestación de contenedores en un entorno local.

El proyecto se ha diseñado con los siguientes objetivos clave:

- Automatización del despliegue mediante Terraform y Kubernetes.
- Escalabilidad de los servicios con Kubernetes.
- Seguridad y segmentación de la red utilizando VPCs y Security Groups en AWS.
- Monitoreo y gestión de incidentes con Amazon CloudWatch.
- Persistencia de datos mediante Amazon S3 y almacenamiento en Kubernetes.

Todos los archivos de configuración, código y scripts han sido organizados en un repositorio de GitHub

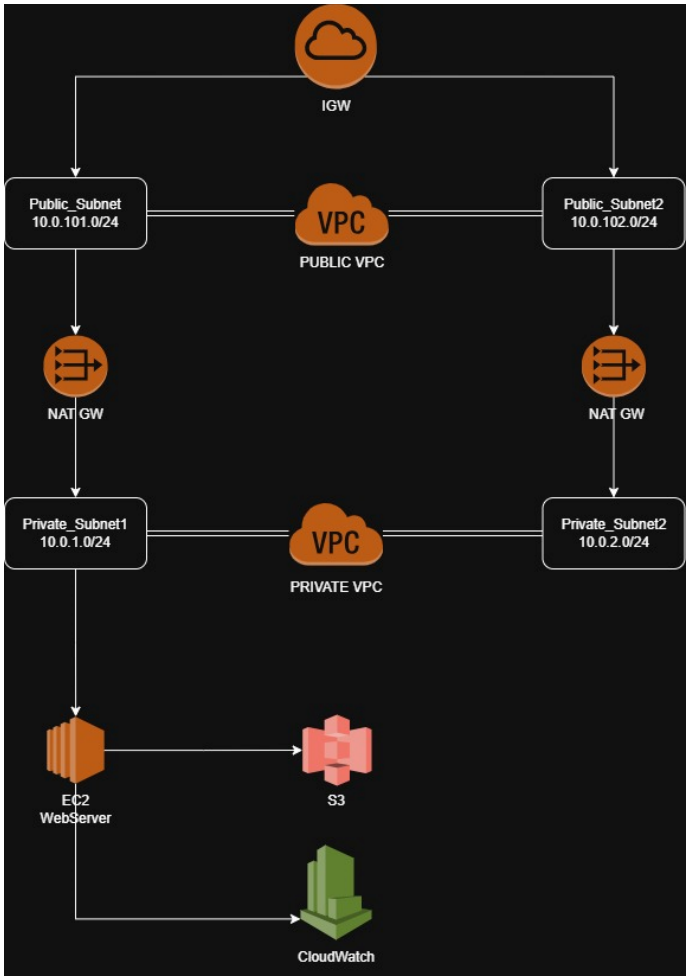
2. Infraestructura en AWS

La infraestructura en AWS se ha diseñado para proporcionar seguridad, escalabilidad y alta disponibilidad en el despliegue de la aplicación web. Se han implementado los siguientes componentes clave:

- Amazon VPC: Se creó una VPC con subredes públicas y privadas para segmentar los recursos de manera segura.
- Subredes Públicas y Privadas:
    - Las subredes públicas (10.0.101.0/24 y 10.0.102.0/24) permiten el acceso a internet mediante una Internet Gateway (IGW).
    - Las subredes privadas (10.0.1.0/24 y 10.0.2.0/24) alojan los servidores EC2 y acceden a internet mediante un NAT Gateway, sin exponerse directamente.
- Instancia EC2 (Web Server):
    - Implementada en una subred privada para mayor seguridad.

    - Configurada con permisos de IAM para interactuar con S3 y CloudWatch.
    - Ejecuta Nginx y permite el almacenamiento de backups en S3.
- Amazon S3: Bucket S3 configurado para almacenar los backups generados por la instancia EC2.
- Amazon CloudWatch:
    - Configurado para monitorear el uso de CPU y otros eventos críticos en la instancia EC2.
    - Se integró con SNS (Simple Notification Service) para enviar alertas en caso de incidencias.
- Security Groups: Configuración de reglas de tráfico seguro para restringir el acceso a la instancia EC2 y otros servicios.

Diagrama:



El diagrama representa la arquitectura de AWS utilizada en el despliegue de la infraestructura. A continuación, se detallan sus componentes:

1. Internet Gateway (IGW): Permite que las subredes públicas tengan acceso a internet.
2. VPC Pública:
    - Contiene subredes públicas (10.0.101.0/24 y 10.0.102.0/24) que pueden acceder directamente a internet a través del IGW.
    - Cada subred pública está conectada a un NAT Gateway para permitir que las subredes privadas tengan acceso a internet de forma segura.
3. VPC Privada:
    - Contiene subredes privadas (10.0.1.0/24 y 10.0.2.0/24), donde se aloja la instancia EC2 sin acceso directo desde internet.
4. Instancia EC2 (Web Server):
    - Se encuentra en una subred privada y ejecuta una aplicación web con Nginx.
    - Se encarga de almacenar backups en Amazon S3.
    - Está configurada para enviar métricas de rendimiento a CloudWatch.
5. Amazon S3: Almacena los backups generados por la instancia EC2.
6. CloudWatch: Monitorea el estado del servidor EC2 y genera alertas en caso de eventos críticos.

Esta arquitectura garantiza una implementación segura y eficiente, minimizando riesgos al restringir el acceso público a la instancia EC2 y optimizando la conectividad de los servicios.

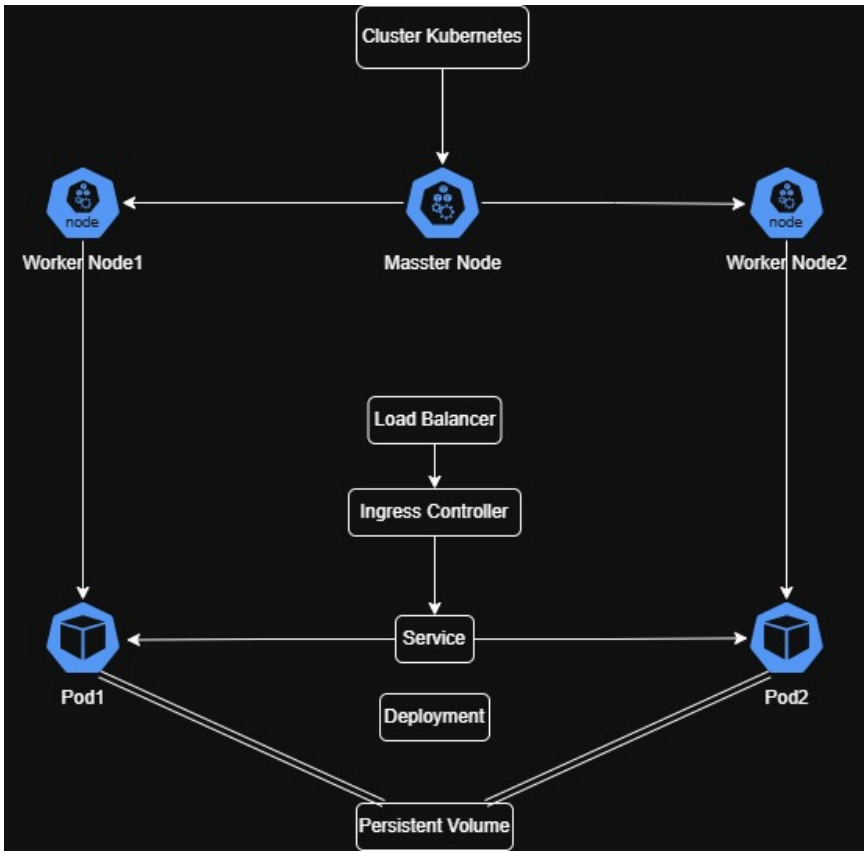
3. Infraestructura en Kubernetes

El despliegue de Kubernetes se realizó en un entorno local utilizando Minikube. Para garantizar escalabilidad y disponibilidad, la aplicación web se implementó mediante un Deployment con 2 réplicas, lo que permite distribuir la carga entre múltiples Pods y proporcionar tolerancia a fallos en caso de que uno de ellos falle.

Para exponer la aplicación a través de la red, se configuró un Service de tipo LoadBalancer, el cual distribuye las solicitudes entre los Pods activos. Además, se implementó un Ingress Controller, que gestiona el acceso externo y aplica reglas de enrutamiento, permitiendo que las solicitudes HTTP/S se dirijan correctamente a los servicios internos del clúster.

Para el almacenamiento persistente de datos, se configuró un Persistent Volume (PV), asegurando que los datos se mantengan disponibles incluso si los Pods se reinician o escalan dinámicamente.

Diagrama:



El diagrama representa la arquitectura del clúster de Kubernetes en el entorno local. Se compone de los siguientes elementos clave:

1. Master Node: Es el nodo de control que administra el clúster de Kubernetes. Gestiona el escalado de los Pods y la distribución del tráfico.
2. Worker Nodes (Worker Node1 y Worker Node2): Son los nodos donde se ejecutan los Pods que contienen la aplicación.
3. Pods: Cada Pod ejecuta una instancia de la aplicación web y está replicado para alta disponibilidad.
4. Load Balancer: Distribuye el tráfico entrante y balancea la carga entre los diferentes Pods.
5. Ingress Controller: Controla las reglas de acceso externo y enruta las solicitudes HTTP/S a los servicios internos.

6. Service: Exposición de la aplicación a través de un punto de acceso dentro del clúster.
7. Deployment: Define la cantidad de réplicas y gestiona la creación y mantenimiento de los Pods.
8. Persistent Volume: Proporciona almacenamiento persistente para los Pods, asegurando que los datos se mantengan disponibles incluso si los Pods se eliminan o escalan.

Con esta arquitectura, se garantiza una aplicación escalable, tolerante a fallos y eficiente en la distribución del tráfico.

4. Seguridad y Escalabilidad

Para garantizar la seguridad de la infraestructura, se han aplicado medidas tanto en AWS como en Kubernetes:

- AWS:
    - Se han configurado Security Groups restrictivos que limitan el acceso a los recursos según la necesidad.
    - La instancia EC2 se ubica en subredes privadas, asegurando que no sea accesible directamente desde internet.
    - Se utiliza un NAT Gateway para permitir que la instancia privada acceda a internet de manera segura.
    - Los roles de IAM restringen los permisos de cada servicio, aplicando el principio de menor privilegio.
- Kubernetes:
    - Se han utilizado Deployments con múltiples réplicas, permitiendo escalabilidad horizontal automática según la carga.
    - El Ingress Controller gestiona las conexiones externas, asegurando un acceso controlado.
    - Se implementó un Persistent Volume para garantizar la persistencia de datos en caso de reinicio o replicación de Pods.

La arquitectura permite alta disponibilidad mediante la combinación de balanceadores de carga en Kubernetes y la segmentación en múltiples subredes en AWS.

5. Monitoreo y Alertas

Para supervisar el estado de la infraestructura y detectar incidentes de manera proactiva, se ha implementado monitoreo en AWS con CloudWatch:

- Métricas clave monitoreadas:
    - Uso de CPU en instancias EC2.
    - Eventos de red y tráfico en la VPC.
- Configuración de alertas:
    - Se han creado alarmas en CloudWatch que detectan umbrales críticos, como un uso de CPU superior al 70%.
    - Las alertas están integradas con Amazon SNS, enviando notificaciones por correo electrónico para una respuesta rápida.

6. Conclusión

Este desafío permitió diseñar e implementar una infraestructura segura, escalable y automatizada en AWS y Kubernetes. Se utilizaron herramientas de Infrastructure as Code (IaC) para garantizar eficiencia y reproducibilidad, junto con monitoreo proactivo para la gestión de incidentes.

La solución final cumple con los requisitos del desafío, asegurando rendimiento, alta disponibilidad y seguridad en el despliegue de la aplicación.

7. Documentación Utilizada

Infraestructura como Código (IaC)

- Terraform Documentation
- Terraform Best Practices

Seguridad en AWS

- AWS Security Groups
- AWS IAM Best Practices

Monitoreo y Gestión de Incidentes

- AWS CloudWatch Alarms
- AWS SNS (Simple Notification Service)

Kubernetes y Escalabilidad

- Kubernetes Horizontal Pod Autoscaler (HPA)
- Kubernetes Deployments